Software Process in Geant4

Gabriele Cosmo (CERN, Geneva, Switzerland)
for the GEANT4 *Collaboration.*

Abstract

Since its erliest years of R&D [1], the GEANT4 simulation toolkit has been developed following software process standards which dictated the overall evolution of the project. The complexity of the software involved, the wide areas of application of the software product, the huge amount of code and Category complexity, the size and distributed nature of the Collaboration itself are all ingredients which involve and correlate together a wide variety of software processes. Although in "production" and available to the public since December 1998, the GEANT4 software product [2] includes Category Domains which are still under active development. Therefore they require different treatment also in terms of improvement of the development cycle, system testing and user support. This article is meant to describe some of the software processes as they are applied in GEANT4 for both development, testing and maintenance of the software.

Keywords: GEANT4, PSS-05, SPICE, Booch, UML

1 Introduction

Many software processes are addressed in GEANT4 arising from various process categories: primary life-cycle of software development, supporting life-cycle, management processes, organisational life-cycle and user-supplier processes. Tailoring of processes is sometimes required because of quality, of stability requirements, or due to the evolution phase of a specific domain, or in order to adapt a process to the people. Software Process Improvement (SPI) is a process which must be applied with the full support of all parties concerned, gradually and after identifying the right priorities and objectives [8]. In this paper we will mention a set of software processes relevant for GEANT4; most of the related procedures and methods of application were already effective during the development phase of the project [1].

2 Primary life-cycle processes

The life-cycle model adopted for most domains in GEANT4 is both iterative and incremental (also called *spiral* approach) [9]. The steps among Analysis of Requirements, Design, Implementation and Testing are repeated. Refinements and extensions to the design have been applied according to new requirements or performance issues. In the current Production and Maintenance phase, the life-cycle model is iterative for most domains.

- **Requirements elicitation process**
  Problem domain and use-case analysis led to elicitation of the User Requirements [10] during the initial phase of the project. User Requirements have been systematically reviewed and updated following the ESA PSS-05 software engineering standard [11]. The User Requirements Document (URD) is now maintained in a source repository providing also automatic versioning; it will be subject to revision during year 2001. Sources of the URD for specific project domains are also kept and maintained in the repository and available to members of the Collaboration.
- **Software Design**
  The Booch (Unified) [9] methodology has been employed for the Object-Oriented Analysis and Design of the software. The Booch/UML notation was chosen as the common *language* for documentation of designs and internal design reviews. The Rational Rose CASE tool [12] has been extensively used for the initial generation of the design documents and, where required, for reverse engineering. A correct domain decomposition and

a well considered set of dependencies (avoiding circular use relationships), allow working groups associated to each Category domain to work largely in parallel, establishing also a hierarchy for delivery. Standard documents provided for architectural and detailed design are: the class *Category Diagram*, *Class Diagrams*, *Scenario Diagrams* and *Class Specifications*.

- **Software Construction**
  Programming and coding guidelines [13] were adopted since the beginning. It was felt important not to impose too fixed rules or style-conventions, but just flexible and adequate guidelines basically dealing with adhesion to the object oriented paradigm (data-hiding, encapsulation, etc.), performance, and portability of the software. Packaging of the software strictly follows the domain decomposition into Categories and sub-Categories which resulted from the design process. Wherever applicable, classes defining interfaces are packaged in sub-Categories separately from the concrete classes implementing such interfaces. In such a way, classes beloging to a Category collaborate to provide a set of services in a re-usable way.
- **Software Integration and Unit Testing**
  System aggregates that can be tested together are identified according to the dependency structure of Categories. Related tests are regularly monitored as part of the routine testing procedures [17]. Unit testing is performed indipendently within each Category or sub-Category, trying to maximize coverage as much as possible.
- **System Testing, Acceptance and Releasing**
  System Testing activity is deployed by a specialised team, the System Testing Team (STT). Procedures for testing [17] and releasing [18] are defined and strictly applied. The release procedure foresees that tags for the various Categories are submitted in groups where the order follows the dependency structure defined by the class Category Diagram. Acceptance tests which are also included in routine system tests are built and run separately by the Release Manager during the release phase. Public releases are scheduled twice a year at fixed dates. Bug-fixes are collected and periodically made available as public patches or minor releases.
- **Software Maintenance**
  In order to achieve maintainable software and ensure its quality, the adoption of standards, wherever possible, is promoted. Encapsulation of components is maximised, interdependency and unit complexity is minimised. We try to assure portability of the software by constantly monitoring the evolution of compilers on different system architectures, and by avoiding adoption of system-dependent solutions or not portable language features. Traceability of updates, extensions and bug-fixes to the code are assured by means of maintaining ad-hoc history files, regularly tagging the code and by trying to disentangle routine development from bug-fix updates [18].
- **User Support, Distribution**
  The terms of the User Support in GEANT4 are defined in the article 2 of the Memorandum of Understanding (MoU) [3] document. Contact persons for each working group are nominated and are responsible for managing and resolving problem reports submitted by users through the WWW by means of the GEANT4 Problem Tracking System [24], based on a customised version of Bugzilla [26]. The GEANT4 WWW site [2] also provides on-line documentation, a FAQ page and the list of contact persons for each Working Group domain. A public User Forum based on Hypernews [25] has been recently set up.

3 Supporting life-cycle processes

- **Documentation**
  As user documentation [19], GEANT4 provides six documents (available on-line from WWW) addressing inherently different topics and levels of expertise. User examples distributed with the toolkit are referenced in the documentation in form of a self-tutorial, with different levels of detail from "novice" to "advanced" applications. A Training Kit Tutorial, documents, papers, publications and much more is also available from the GEANT4 WWW site [2].
- **Configuration and Change Management**
  - *Software Configuration Management*: A server for software and documentation repositories is in place; it is based on CVS [20] as basic tool for concurrent version management. The code and documents in the repositories are accessible to members of the Collaboration through AFS [21] at CERN and also through "pserver" read/write access.
  - *Tagging and Versioning*: Category coordinators have the responsibility of managing the development within their Categories and provide tags for testing and releases following well specified rules [18]. The STT will then run the system tests for the supported architectures. The Bonsai [26] tool is used as database to automatically detect any new tag introduced into CVS. A global reference tag is provided every month, including all tags which passed system validation tests. The tag is announced and made available to developers and collaborating Institutes for development. User documentation is tagged according to the public releases.
- **Quality Assurance and Measurement**
  Code walk-throughs are periodically performed through specialized tools for monitoring against violations of established coding rules. The CodeWizard tool [14] is used; an automatic mechanism for submitting code filtering for unit Categories has been put in place and is available to GEANT4 developers. Checks on run-time memory management are regularly performed before each public release; tools like Insure++ [15] and SUN Workshop [16] are used over selected test-bed applications. Checks for violations of the dependency structure of Categories at macro level are performed periodically and correspondance with the main class Category Diagram is monitored. Performance monitoring on selected test-bed applications is applied at unit level for Categories where performance is critical.
- **Verification and Validation**
  At macro level, the dependency structure of Categories is verified against the main class Category Diagram taken as reference. General functionalities of the Toolkit are verified at every new revision of the URD [10]. Verifications of functionalities and coverage at unit level are under responsibility of each Category Coordinator, as well as unit tests and validation of new developments or fixes. New development is validated by the STT once all system integration tests have been successfully performed, provided that, in collaboration with Category Coordinators, system tests have been extended to cover also the new introduced functionalities.

4 Organisational life-cycle processes

- **Project Tasks Management**
  Management of the project is under control of the GEANT4 Collaboration as specified and dictated in the GEANT4 Memorandum of Understanding [3]. Geant 4 is organised in working groups; each working group is responsible for one specific domain of the toolkit

where well specified tasks are associated to it. Each working group is represented in the TSB by its Coordinator(s). Objectives are defined every year, discussed within the TSB meeting and presented to the CB. They are reviewed generally during the TSB meetings, joint reviews or in the Collaboration Workshop organised once every year.

- **Improvement Process**
  A plan for SPI [4] was formally presented at TSB meetings and approved as one of the milestones for year 2000-2001. Priorities and objectives were identified and the approved SPI program is currently being applied. Formal assessments based on the exemplar model ISO-15504 (SPICE) [6] are performed. SPI is considered life-cycle driven, therefore progresses of the established program are constantly monitored. Reaching the capability level of an established process [5] in the project is a key objective for GEANT4.

References

[1] S.Giani et al., *GEANT 4 : An Object-Oriented Toolkit for Simulation in HEP*, CERN/LHCC/98-44, November 1998.

[2] `http://cern.ch/geant4`.

[3] `http://cern.ch/geant4/organisation/MOU.html`.

[4] `http://cern.ch/geant4/milestones/software\_process/OOAD-items.html`.

[5] ISO/IEC Joint Technical Committee 1 (JTC1), *ISO/IEC DTR 15504 Software Process Assessment*, Project Editor: Terry Rout.

[6] ISO/IEC Joint Technical Committee 1 (JTC1), *ISO/IEC DTR 15504-5 Part 5: An Assessment Model and Indicator Guidance*, Product Editor: Jean Martin Simon.

[7] M.Paulk et al., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison-Wesley Publishing Co., 1995, ISBN 0-201-54664-7.

[8] D.A.Reo et al., *Measuring Software Process Improvement: there's more to it than just measuring processes*, ESI - FESMA 99, September 1999.

[9] G.Booch, *Object-Oriented Analysis and Design with Applications*, The Benjamin/Cummings Publishing Co., 1994, ISBN 0-805-35340-2.

[10] RD44, *GEANT 4 User Requirements Document*, CERN, 1998.

[11] ESA, *Guide to User Requirements Definition Phase*, ESA PSS-05, 1994.

[12] I.White and M.Goldberg, *Using the Booch Method, a Rational Approach*, Rational Rose Corp., The Benjamin/Cummings Publishing Co., 1994, ISBN 0-805-30614-5.

[13] `http://cern.ch/geant4/collaboration/coding\_guidelines.html`.

[14] `http://www.parasoft.com/products/wizard/index.htm`.

[15] `http://www.parasoft.com/products/insure/index.htm`.

[16] `http://www.sun.com/forte/cplusplus/`.

[17] `http://cern.ch/geant4/working\_groups/testing/testing.html`.

[18] `http://cern.ch/geant4/working\_groups/testing/tag\_release\_policy.html`.

[19] `http://cern.ch/geant4/G4UsersDocuments/Overview/html/index.html`.

[20] P.Cederqvist et al., *Version Management with CVS*, Signum Support AB, 1992.

[21] *The AFS User Guide*, IBM Transarc Corp., April 2000.

[22] G.Cosmo, G.Pawlitzek and H.P.Wellisch, GEANT4 *Design Process Assessment*, Internal Note, September 2000.

[23] `http://cern.ch/geant4/working\_groups/testing/testing.html`.

[24] `http://wwwinfo.cern.ch/asd/cgi-bin/geant4/problemreport/`.

[25] `http://geant4-hn.slac.stanford.edu:5090/Geant4-HyperNews/index`.

[26] `http://bonsai.mozilla.org/`, `http://bugzilla.mozilla.org/`, `http://tinderbox.mozilla.org/`.

[27] `http://lxr.linux.no/`.